




清华师哥封神之作！RocketMQ核心笔记疯传Ali内网

原创

普通网友  于 2021-07-26 13:55:13 发布  35  收藏

分类专栏：[面试题](#) 文章标签：[java](#)

版权声明：本文为博主原创文章，遵循 [CC 4.0 BY-SA](#) 版权协议，转载请附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/java_dazhuzhu/article/details/119107791

版权



[面试题](#) 专栏收录该内容

96 篇文章 1 订阅

订阅专栏

消息队列（RocketMQ）作为高并发系统的核心组件之一，能够帮助业务系统解构提升开发效率和系统稳定性。

RocketMQ 是一款分布式、队列模型的消息中间件，具有以下特点：

- 能够保证严格的消息顺序
- 提供丰富的消息拉取模式
- 高效的订阅者水平扩展能力
- 实时的消息订阅机制
- 亿级消息堆积能力

选择**RocketMQ**的理由：

- 强调集群模式无单点，可扩展，任意一点高可用，水平扩展
- 海量数据的堆积能力，消息堆积后，写入延迟低
- 支持上万个队列
- 消息失败重试机制
- 消息可查询
- 开源社区灵活
- 成熟度（支持阿里双十一）

RocketMQ 核心的四大组件：Name Server、Broker、Producer、Consumer 每个组件都可以部署成集群模式进行水平扩展。

如果搞懂这份RocketMQ实战与原理解析，在简历上写精通RocketMQ完全没问题

由于篇幅原因，这份**RocketMQ**核心实战原理解析已经被整理成了PDF文档，有需要这份**RocketMQ**核心实战原理解析完整文档的麻烦转发后私信回复“666”即可获取资料免费领取方式！

阿里、滴滴等资深专家对**RocketMQ**实战与原理解析的评价

从2013年开源至今，RocketMQ承载了阿里巴巴数年“双11”的峰值流量，并且被业界多个大型互联网公司、大型央企和金融证券系统广泛使用。本书从开发和运维的双重视角对RocketMQ做了详细的阐述，既能满足入门读者的需求，又能满足需要通过源代码了解RocketMQ工作原理的中高端读者的需求。

—— **王小瑞** 阿里巴巴资深技术专家/Apache RocketMQ PMC Chair

这是一本非常具有实战意义的手册，可以帮助工程师快速了解RocketMQ并展开实操。理论清晰，案例实用，体现了作者深厚的技术功底。

—— **夏振宇** 微瑞思创董事长

消息中间件是分布式系统中依赖最广泛的中间件产品，作为Apache中间件顶级项目，RocketMQ已经经历了众多大型互联网公司的线上检验，不论是从可靠性还是吞吐量上都得到广泛的认可。相信这本书的出版，对正在使用和计划研究RocketMQ技术的开发者来说是个大大的福音。

—— **王晓东** 凤凰金融高级副总裁

消息队列是重要的中间件之一，已经成为大型应用不可或缺的组件。本书从原理和应用的角度对RocketMQ进行了详细的讲解，无论是入门还是进阶，本书都可以作为你的良师益友。

—— **张彦龙** 滴滴出行高级数据专家

作者在RocketMQ领域有多年的前线开发、调优经验，他将以独到的方式带领你走上RocketMQ的进阶之路。本书可以帮助开发者，更加高效、快速地构建起分布式服务，将工程师从服务稳定性、分布式事务一致性的桎梏中解放出来。

—— **耿嘉安** 360大数据专家

注：需要RocketMQ核心笔记【[点击此处即可免费领取](#)】

RocketMQ实战目录展示

RocketMQ实战原理解析从入门到生产环境、分布式消息列队、可靠性、吞吐量、Apache中间件、NameServer源码、主从同步，最后基于Netty的通信实现

第1章 快速入门	5
第2章 生产环境下的配置和使用	12
第3章 用适合的方式发送和接收消息	27
第4章 分布式消息队列的协调者	49
第5章 消息队列的核心机制	61
第6章 可靠性优先的使用场景	68
第7章 吞吐量优先的使用场景	78
第8章 和其他系统交互	92
第9章 首个Apache中间件顶级项目	101
第10章 NameServer源码解析	107
第11章 最常用的消费类	116
第12章 主从同步机制	132
第13章 基于Netty的通信实现	139

1、快速入门

本篇幅可以让你了解RocketMQ和分布式消息队列的功能，然后搭建好单机版的消息队列，进而能够发送并接收简单的消息。

- 消息队列功能介绍

- RocketMQ简介
- 快速上手RocketMQ

1.1	消息队列功能介绍	1
1.1.1	应用解耦	1
1.1.2	流量消峰	2
1.1.3	消息分发	3
1.2	RocketMQ 简介	4
1.3	快速上手 RocketMQ	4
1.3.1	RocketMQ 的下载、安装和配置	5
1.3.2	启动消息队列服务	6
1.3.3	用命令行发送和接收消息	6
1.3.4	关闭消息队列	6
1.4	本章小结	7

阿里的消息中间件有很长的历史，从 2007 年的 Notify 到 2010 年的 Napoli，2011 年升级后改为 MetaQ，然后到 2012 年开始做 RocketMQ，RocketMQ 使用 Java 语言开发，于 2016 年开源。第一代的 Notify 主要使用了推模型，解决了事务消息；第二代的 MetaQ 主要使用了拉模型，解决了顺序消息和海量堆积的问题。RocketMQ 基于长轮询的拉取方式，兼有两者的优点。

每一次产品迭代，都吸取了之前的经验教训，目前 RocketMQ 已经成为 Apache 顶级项目。在阿里内部，RocketMQ 很好地服务了集团大小上千个应用，在每年的双十一当天，更有不可思议的万亿级消息通过 RocketMQ 流转（在 2017 年的双十一当天，整个阿里巴巴集团通过 RocketMQ 流转的线上消息达到了万亿级，峰值 TPS 达到 5600 万），在阿里大中台策略上发挥着举足轻重的作用。

此外，RocketMQ 是使用 Java 语言开发的，比起 Kafka 的 Scala 语言和 RabbitMQ 的 Erlang 语言，更容易找到技术人员进行定制开发。

2、生产环境下的配置和使用

在生产环境中使用 RocketMQ 集群需要比 QuickStart 部分了解更多的内容，本章节在机器角色、集群配置和部署，以及集群管理方面都做了介绍，用户可以基于这些内容搭建起一个生产环境的 RocketMQ 消息队列集群，在数据量不大的非关键场景，可以通过这一章节快速上线。

- RocketMQ 各部分角色介绍

- 多机集群配置和部署
- 发送、接收消息示例
- 常用管理命令
- 通过图形界面管理集群

2.2	多机集群配置和部署.....	9
2.2.1	启动多个 NameServer 和 Broker	10
2.2.2	配置参数介绍	11
2.3	发送 / 接收消息示例	13
2.4	常用管理命令	15
2.5	通过图形界面管理集群	21
2.6	本章小结	22

3、用适合的方式发送和接收消息

对消息队列使用者来说，Consumer和Producer是打交道最多的两个类型。

本篇详细介绍了两种类型的Consumer和一种类型的Producer,用户在使用的时候基于业务需求来选择合适的类型。最后重点介绍了Offset和Log，了解Offset机制是正确使用RocketMQ的基础，合理使用Log可以大幅提高开发、调试的效率。

- 不同类型的消费者
- 不同类型的生产者
- 如何存储队列位置信息
- 自定义日志输出

第3章 用适合的方式发送和

接收消息 23

3.1 不同类型的消费者 23

3.1.1 DefaultMQPushConsumer 的 使用 23

3.1.2 DefaultMQPushConsumer 的 处理流程 25

3.1.3 DefaultMQPushConsumer 的 流量控制 28

3.1.4 DefaultMQPullConsumer 30

3.1.5	Consumer 的启动、关闭	
	流程	32
3.2	不同类型的生产者	33
3.2.1	DefaultMQProducer	34
3.2.2	发送延迟消息	36
3.2.3	自定义消息发送规则	36
3.2.4	对事务的支持	37
3.3	如何存储队列位置信息	38
3.4	自定义日志输出	42
3.5	本章小结	44

4、分布式消息队列的协调者

介绍了NameServer的功能，NameServer在RocketMQ集群中扮演调度中心的角色。

各个Producer、Consumer 上报自己的状态上去，同时从NameServer获取其他角色的状态信息。NameServer的功能虽然非常重要，但是被设计得很轻量级，代码量少并且几乎无磁盘存储，所有的功能都通过内存高效完成。

还介绍了底层的通信机制，RocketMQ基于Netty对底层通信做了很好的抽象，使得通信功能逻辑清晰，代码简单。

- NameServer的功能
- 各个角色间的交互流程
- 底层通信机制

4.1	NameServer 的功能	45
4.1.1	集群状态的存储结构	46
4.1.2	状态维护逻辑	47
4.2	各个角色间的交互流程	48
4.2.1	交互流程源码分析	48
4.2.2	为何不用 ZooKeeper	50
4.3	底层通信机制	50
4.3.1	Remoting 模块	51
4.3.2	协议设计和编解码	54
4.3.3	Netty 库	56
4.4	本章小结	56

5、消息队列的核心机制

介绍了RocketMQ消息队列实现的难点及核心，即“队列”本身的实现，基于磁盘做一个读写效率高的队列并非易事，实现不好就会使磁盘操作成为整个系统的瓶颈，无法提升系统的吞吐量。RocketMQ基于“顺序写”“随机读”的原则来设计，利用“零拷贝”技术，克服了磁盘操作的瓶颈。

另一个难点是为了高可用性而设计的主从机制，数据被及时复制到多个机器，这样当一台机器出故障后，整体系统依然可用。这样可靠性和性能直接有个权衡，RocketMQ 把选择权留给用户，用户根据具体的业务场景来选择要更高的可靠性，还是要更高的效率。

- 消息存储和发送
- 消息存储结构
- 高可用性机制
- 同步刷盘和异步刷盘
- 同步复制和异步复制

5.2	消息存储结构.....	58
5.3	高可用性机制.....	60
5.4	同步刷盘和异步刷盘.....	61
5.5	同步复制和异步复制.....	62
5.6	本章小结.....	63

6、可靠性优先的使用场景

本篇根据使用场景，讨论如何“可靠”地收发消息。即在要求消息顺序的场景下，如何既能并发执行，又能保证消息顺序；然后分析可能的故障场景下，如何应对以保证不丢消息、不中断服务。RocketMQ 在设计上，有重试机制来保证消息不丢，造成的结果是可能存在消息重复，这一点需要用户根据具体业务场景来处理。下一章将讨论处理大数据量消息的方法。

- 顺序消息
- 消息重复问题
- 动态增减机器
- 各种故障对消息的影响
- 消息优先级

6.1	顺序消息	64
6.1.1	全局顺序消息	64
6.1.2	部分顺序消息	65
6.2	消息重复问题	67
6.3	动态增减机器	67
6.3.1	动态增减 NameServer	67
6.3.2	动态增减 Broker	69
6.4	各种故障对消息的影响	70
6.5	消息优先级	72
6.6	本章小结	73

7、吞吐量优先的使用场景

本篇重点关注性能，关注在大消息量的情况下，如何提高RocketMQ的吞吐量。首先介绍了消息过滤，在服务端进行消息过滤可以减少无效消息传输造成的带宽浪费，Tag是最常用的一种高效过滤方式，此外还可以用SQL表达式、FilterServer来过滤消息。

另一个提高吞吐量的方法是增加集群的机器数量，提高并发性，要根据实际场景增加Broker、Consumer 或Producer角色的机器数量。

- 在Broker端进行消息过滤

- 提高Consumer处理能力
- Consumer的负载均衡
- 提高Producer的发送速度
- 系统性能调优的一般流程

第7章 吞吐量优先的使用场景……74

7.1	在 Broker 端进行消息过滤 ……	74
7.1.1	消息的 Tag 和 Key ……	74
7.1.2	通过 Tag 进行过滤 ……	75
7.1.3	用 SQL 表达式的方式进行 过滤 ……	75
7.1.4	Filter Server 方式过滤 ……	77
7.2	提高 Consumer 处理能力 ……	78
7.3	Consumer 的负载均衡 ……	80

7.3.1	DefaultMQPushConsumer 的 负载均衡 ……	80
-------	------------------------------------	----

7.3.2	DefaultMQPullConsumer 的 负载均衡 ……	81
-------	------------------------------------	----

7.4	提高 Producer 的发送速度 ……	83
-----	----------------------	----

7.5	系统性能调优的一般流程 ……	85
-----	----------------	----

7.6	本章小结 ……	87
-----	---------	----

由于篇幅原因，这份RocketMQ核心实战原理解析已经被整理成了PDF文档，有需要这份RocketMQ核心实战原理解析完整文档的麻烦转发后私信回复“666”即可获取资料免费领取方式!

8、和其他系统交互

作为一个中间件产品，会比普通软件更多地需要和其他系统打交道，本篇介绍了如何与SpringBoot、Spark、Flink等软件进行交互。同时介绍了使用云端的RocketMQ产品，以及自定义开发运维工具的方法。

- 在SpringBoot中使用RocketMQ
- 直接使用云上RocketMQ
- RocketMQ与Spark、Flink对接
- 自定义开发运维工具

8.1	在 SpringBoot 中使用	
	RocketMQ	88
8.1.1	直接使用	88
8.1.2	通过 Spring Messaging 方式 使用	90
8.2	直接使用云上 RocketMQ	91
8.3	RocketMQ 与 Spark、Flink 对接	93
8.4	自定义开发运维工具	93
8.4.1	开源版本运维工具功能 介绍	94
8.4.2	基于 Tools 模块开发自定义 运维工具	95
8.5	本章小结	96

9、首个Apache中间件顶级项目

RocketMQ是阿里最优秀的中间件之一，本篇介绍了RocketMQ的历史，以及其目前作为Apache顶级项目的现状。

- RocketMQ的前世今生
- Apache顶级项目(TLP)之路

- 源码结构
- 不断迭代的代码

2016 年，MetaQ 在双十一期间承载了万亿级消息的流转，跨越了一个新的里程碑，同时 RocketMQ 进入 Apache 孵化。



图 9-1 RocketMQ 演进历史

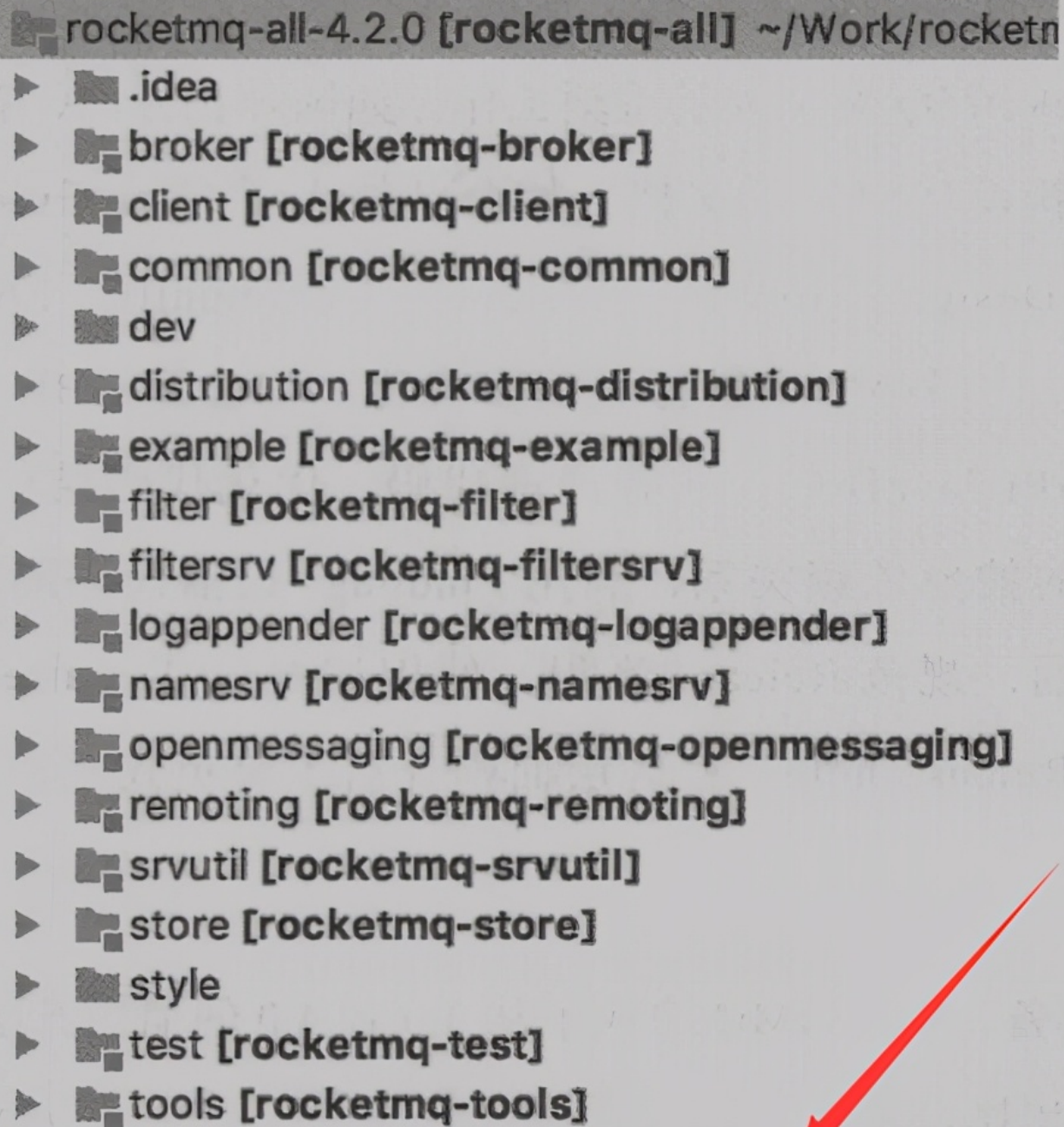


图 9-2 RocketMQ 源码结构

10、NameServer源码解析

本篇分析了NameServer模块的源码，NameServer是一个功能重要但是代码量不大的模块，所以选择这个模块入手，比较容易理解。我们在分析源码时，认真读懂一个模块后就可以对作者的代码风格、设计偏好等有基本的了解。

- 模块入口代码的功能
- NameServer的总控逻辑
- 核心业务逻辑处理
- 集群状态存储

第 10 章 NameServer 源码

解析	103
10.1 模块入口代码的功能	103
10.1.1 入口函数	103
10.1.2 解析命令行参数	104
10.1.3 初始化 NameServer 的 Controller	105
10.2 NameServer 的总控逻辑	106
10.3 核心业务逻辑处理	107
10.4 集群状态存储	109
10.5 本章小结	111

11、最常用的消费类

本篇分析的是Client模块里的代码，我们在使用RocketMQ的时候，更多的是和这个模块里的代码打交道。本章重点分析了DefaultMQPushConsumerImpl类，然后分析了Consumer的并发处理过程，最后分析了客户端Class统一的底层通信类MQClientInstance。

- 整体流程
- 消息的并发处理。
- 生产者，消费者的底层类

11.1 整体流程	112
11.1.1 上层接口类	112
11.1.2 DefaultMQPushConsumer 的实现者	114
11.1.3 获取消息逻辑	116
11.2 消息的并发处理	118
11.2.1 并发处理过程	118
11.2.2 ProcessQueue 对象	121
11.3 生产者消费者的底层类	122
11.3.1 MQClientInstance 类的 创建规则	122

12、主从同步机制

本篇分析了Master和Slave角色的Broker之间同步信息功能的实现。需要同步的信息分为两种类型，实现方式各不相同：一种是元数据信息，采用基于Netty的command方式来同步消息；另一种是commitLog信息，同步方式是直接基于JavaNIO来实现。

- 同步属性信息
- 同步消息体
- `sync_master` 和 `async_master`

12.1	同步属性信息.....	128
12.2	同步消息体.....	130
12.3	<code>sync_master</code> 和 <code>async_master</code> ...	132
12.4	本章小结.....	134

13、基于Netty的通信实现

本篇介绍了RocketMQ底层通信的实现机制，由于它是基于Netty来实现的，所以首先介绍了Netty的基础知识。Netty被用在很多开源软件的底层通信部分，RocketMQ以Netty为基础，还实现了一种机制，把通信功能和消息处理功能分离，不同类型的通信内容被抽象成发送带有对应类型代码的Command，同时根据类型代码查找对应的Processor和Executor来执行，结构非常清晰，为我们自己实现网络通信程序提供了参考。

- **Netty**介绍
- **Netty**架构总览
- **Netty**用法示例
- **RocketMQ**基于**Netty**的通信

第 13 章 基于 Netty 的通信实现... 135

13.1	Netty 介绍.....	135
13.2	Netty 架构总览.....	136
13.2.1	重新实现 ByteBuffer.....	136

13.2.2	统一的异步 I/O 接口.....	137
13.2.3	基于拦截链模式的事件 模型	138
13.2.4	高级组件.....	139
13.3	Netty 用法示例.....	140
13.3.1	Discard 服务器	140
13.3.2	查看收到的数据.....	144
13.4	RocketMQ 基于 Netty 的通信 功能实现.....	145
13.4.1	顶层抽象类.....	145
13.4.2	自定义协议.....	148
13.4.3	基于 Netty 的 Server 和 Client	151
13.5	本章小结.....	152

由于篇幅原因，这份RocketMQ核心实战原理解析已经被整理成了PDF文档，有需要这份RocketMQ核心实战原理解析完整文档的麻烦 一键三连+评论 🍷🍷🍷【[点击此处即可 免费领取](#)】！